AONS As Built Specification

David Levy

Revision History

Revision 2.0.0 beta 20070806	**Monday, 8 August 2007**	**DavidLevy<dlevy@nla.gov.au>**
	Initial Beta Release	

AONS As Built Specification

David Levy

Table of Contents

List of Tables

Chapter 1. Overview

Goals

The *As Built Specification* should address the following set of goals:

- Discuss what AONS is from a users perspective
- Demonstrate how to build and deploy AONS
- Discuss development of AONS
- Discuss methodologies, design patterns or other areas of complexity within AONS which are of interest to a technical audience

One of the hardest parts about writing documentation is that there is essentially a bootstrapping process invovled. Do we begin at the top with what AONS is trying to achieve (see Functional Requirements document) or do we begin at the bottom with details such as project layout. This is hard to disect, but I'll break it down into the following areas:

- What exactly is AONS
- Setting up the project as a developer
- Building/Deploying the application
- Methodologies/Design Patterns
- Technologies Utilised
- Common Tasks

Chapter 2. Functional Requirements

Purpose

This chapter discusses the requirements in regard to what this version of AONS must "do". functional requirements, like requirements for building the project, are specified under ???.

Functional Requirement Breakdown

AONS functional requirements can be thought of as two main groupings: primary and supporting groups. The primary groups are where we concentrate functionality relating to core domain objects. Supporting functional requirements are for those functions which support and facilitate access to the primary areas of functionality.

Primary functional requirements are given below:

Table 2.1. Primary Functional Requirements

Registry Functional requirements	Functional requirements which works with the configuration of external registries of authoritative information used to aid in the judgement of obsolescence.
Repository Functional requirements	The area of the system which will work with repositories of data files which need judgement of their obsolescence.
Internal Format Functional requirements	The part of the system which stores information about formats. This information may be gleaned from external registries or the client's knowledge.
Obsolescence Functional requirements	The part of the system which deals with making judgements on the obsolescence of a format within a data repository.

Supporting functional requirements are given below:

Table 2.2. Supporting Functional Requirements

Task Functional Requirements	This area of the system is the task system which enables a repository manager to view changes in the system and perform work as needed. It's almost a primary area of functionality... but not quite.
Logging Functional Requirements	Deals with logging, separate from server logs, these are intended for use by the installation administrator/repository manager.
Email Functional Requirements	Covers the notification via email from an AONS installation.
RSS Interface Functional Requirements	Covers the notification via RSS/Atom from an AONS installation.

Search/Indexing Subsystem Functional Requirements	Covers the underlying search functionality which the primary modules utilise to save their information.
Http Subsystem Functional Requirements	The underlying subsystem which controls access to remove HTTP servers. This is mentioned since some installations of AONS may configure things like proxy servers and we need a single access point.
Configuration Subsystem Functional Requirements	This is the essence of runtime configuration of AONS without needing to stop and modify XML.
REST Interface Functional Requirements	This is the interface into AONS which is via REST

Registry Functional requirements

Functional requirements which works with the configuration of external registries of authoritative information used to aid in the judgement of obsolescence.

Table 2.3. Registry Functional Requirements

ID	Name	Description
REG1	Create Registry Configuration	Create a new registry configuration (for now: GDFR, PRONOM and LC DFW).
REG2	Retrieve Registry Configuration	Retrieve a registry by the registries ID.
REG3	Update Registry Configuration	Update an existing registry configuration.
REG4	Delete Registry Configuration	Delete an existing registry configuration.
REG5	Create Synchronize Schedule	Create an update schedule dictating when a registry should update itself.
REG6	Retrieve Synchronize Schedule	Retrieve a registry update schedule
REG7	Update Synchronize Schedule	Update synchronize schedule for a registry
REG8	Delete Synchronize Schedule	Delete a synchronize schedule which will in turn mean that the registry will only be updated manually.
REG9	Configure Multiple Registry Merge Algorithm	Configure what mechanism to use when multiple registries have the same information: 1. Preferential Merge (AONS v1.0 munge) 2. Manually Resolve Conflict
REG10	Manually Run Update	

		Run the update for the registry manually.
REG11	Synchronize	All registries should be able to synchronize with their external representation
REG12	Create External Format	As the result of a registry synchronize, it should create any new external formats within the AONS installation not already present
REG13	Update External Format	As the result of a registry synchronize, it should update any changed external formats within the AONS installation out of synchronization with those in the external registry
REG14	Delete External Format	As the result of a registry synchronize, it should delete any changed external formats within the AONS installation which have been deleted externally
REG15	Required Events	Must generate events for the following actions on a Registry: created, updated, deleted, synchronize started and synchronize ended
REG16	Synchronize Event details	Synchronize ended event should indicate the status as well as information about what changed during the synchronize (formats created etc)
REG17	External Change Detection	Synchronize should detect changes in external formats using a hash
REG18	Statistical Data	Statistical data should be kept for Registry changes to allow graphing (formats created, updated, deleted and character count and change amount, total formats)

Repository Functional requirements

The area of the system which will work with repositories of data files which need judgement of their obsolescence.

Table 2.4. Repository Functional Requirements

ID	Name	Description

REP1	Create Repository Configuration	Create a new repository configuration to the system (ORCA, Fedora and D-Space).
REP2	Retrieve Repository Configuration	Retrieve repository configuration by ID.
REP3	Update Existing Repository Configuration	Update an existing repository configuration.
REP4	Delete Existing Repository Configuration	Delete an existing repository configuration.
REP5	Configure Update Schedule	Configure a repository update schedule.
REP6	Manually Run Crawl	Manually re-run a crawl on a data repository.
REP7	Configure Filter	Configure a domain specific filter (DSpace, Fedora etc) for a repository.
REP8	Go to Obsolescence Information for Repository	Jump from Data Repository area of functional requirements to Obsolescence area keeping the focus on this particular Data Repository.
REP9	Required Events	Must generate events for the following actions of a Repository: created, updated, deleted, scan started and scan ended
REP10	New Metadata Event	Must generate an event when a new unique format identification metadata is found as a result of a repository scan
REP11	Configurable Identification	Must allow for configuration of different identification tools (if the repository type allows it), both at a global level and also on a repository by repository basis.
REP12	Temporal/Snapshots	Must maintain a history of previous snapshots for statistical analysis

Internal Format Functional requirements

The part of the system which stores information about formats. This information may be gleaned from external registries or the client's knowledge.

Table 2.5. Internal Format Functional Requirements

ID	Name	Description
INT-FMT1	Create Format	Manually create a new format in the system.

INT-FMT2	Use External Registry as Basis for New Format	Utilise one or more external format definitions as the basis for a new format in the system.
INT-FMT3	Update Format	Manually update format in the system.
INT-FMT4	Notify External Format Changed	Notify the repository owner when an external registry format has changed which was used to derive an internal format.
INT-FMT5	Delete Format	Delete an existing format from the system.
INT-FMT6	Update Format Base on External Changes	Assist the repository owner update their internal concept of a format based on the changes in an external format.
INT-FMT7	Create Internal Formats Based on Repository Crawl	Create “internal” formats based on a data repository crawl. This process should give the users some options, like retrieving template information from a registry, providing an option to notify any conflicts. We should also provide the ability to do this globally (all format ids found in a repository) or for a particular format found in a repository.
INT-FMT8	Work Through Bulk External Conflicts	The system should allow a registry owner to work through bulk changes in external formats in a relatively easy manor.
INT-FMT9	Notify When New Format Found in Data Repository	Notify when a new format is discovered in a data repository. This should be used as an option instead of automatically creating a format based on an external format.
INT-FMT10	Search Known Format Information	Search for format information both in this system and in external registries.
INT-FMT11	View External and Internal Format Details	The system should allow a user to view an external and internal format details.
INT-FMT12	Format Navigation	The system should allow a user to navigate from a main format menu to the following sub-menus: 1. Search Formats

		2. Explore Formats 3. Create Formats 4. View Format Log
INT-FMT13	Utilisation Count Increment	Whenever an AONS format links to a Registry format, it should increase the number of formats linked by one, if that format doesn't already link to a Registry Format within that registry.
INT-FMT14	Utilisation Count Decrement	Whenever an AONS format unlinks from a Registry format, it should decrease the number of formats linked by one, if that format has no remaining AONS formats linking to it

Obsolescence Functional requirements

The part of the system which deals with making judgements on the obsolescence of a format within a data repository.

Table 2.6. Obsolescence Functional Requirements

ID	Name	Description
OBS1	Default Obsolescence Rules	Ensure that when the application is deployed we create/configure a default set of obsolescence rules (FORDDS 1 & 2)
OBS2	View Obsolescence Rules	View the currently configured community and local rules.
OBS3	Update Obsolescence Rules	Update the currently configured obsolescence rule questions.
OBS4	Configure Global Obsolescence Rankings	Modify the different categories of obsolescence and their corresponding numerical values.
OBS5	Produce a Current or Historical Obsolescence Report	Provide an obsolescence snapshot, in either XML, PDF or HTML of the current obsolescence view of the system. This should incorporate all supporting information (formats and software) used to produce the Obsolescence Report. The user may choose to produce a report now or for a past view of the system.
OBS6	Risk Assessment	

		Allow user to perform risk assessments on internal formats
OBS7	Risk Expiry	Set up the system so that risk assessments are only valid for a period then they expire.
OBS8	Risk Expiry Event	Ensure a notification is sent (via an event) to the repository owner when a risk assessment expires
OBS9	Risk Expiry Reminder	Ensure a notification is sent (via an event) to the repository owner prior to expiring a risk assessment

Task Functional Requirements

This area of the system is the task system which enables a repository manager to view changes in the system and perform work as needed. It's almost a primary area of functionality... but not quite.

Table 2.7. Task Functional Requirements

TASK1	Task Creation	Within the system, certain events should create tasks
TASK2	Task Completion	Within the system, certain events should mark tasks as completed tasks
TASK3	Subject Field	Tasks should have a "subject"
TASK4	Message Field	Tasks should have a "message"
TASK5	Creation Date Field	Tasks should have a "date of creation"
TASK6	Completion Date Field	Tasks should have a "date of completion"
TASK7	Status Field	Tasks should have a "status"
TASK8	List of Possible Actions	Tasks should have a "list of possible actions"
TASK	Create Repository Task	We should create a task "Create Repository" upon the events: • There are no repositories Last repository deleted The task should be marked as "completed" by any of the following events: • A repository has been created
TASK	Crawl Repository Task	We should create a task "Crawl Repository" upon the events: • Repository has no previous format scan; and

		Repository has no schedule The task should be marked as "completed" by any of the following events: • Repository has a sucessful format scan
TASK	Create Registry Task	We should create a task "Create Registry" upon the events: • No registries in system Last registry deleted The task should be marked as "completed" by any of the following events: • Registry created
TASK	Syncrhonize Registry Task	We should create a task "Synchronize Registry" upon the events: • Registry has no previous synchronization; and Registry has no schedule The task should be marked as "completed" by any of the following events: • Registry has been synchronized; or Registry has a synchronize schedule
TASK	Identify Format Metadata Task	We should create a task "Identify Format Metadata" upon the events: • The creation of a format identification metadata; or The "unlinking" of a format identification metadata from all internal formats thus making it "unidentified" The task should be marked as "completed" by any of the following events: • The format identification metadata is linked to an internal format

TASK	Review Internal Format Tak	We should create a task "Review Internal Format" upon the events: • New internal format created (and thus has no risk assessment) A previous risk assessment has expired; or A previous risk assessment has reached the reminder period The task should be marked as "completed" by any of the following events: • Risk assessment has been performed

Logging Functional Requirements

Deals with logging, separate from server logs, these are intended for use by the installation administrator/ repository manager.

Table 2.8. Logging Functional Requirements

ID	Name	Description
LOG1	Message Storage	All log messages must be kept indefinitely
LOG2	Severity Attribute	Messages must have a "severity" property
LOG3	Component Name Attribute	Messages must have a "component name" property
LOG4	Subject	Messages must have a "subject" property which is a quick status line about the nature of the event
LOG5	Message	Messages must have a "message" property which embodies the descriptive message of the event
LOG6	Date Created	Messages must have a "date created" property
LOG7	Searching	Messages must be searchable, either by a general search of fields or on a specific property

Email Functional Requirements

Covers the notification via email from an AONS installation.

Table 2.9. Email Functional Requirements

ID	Name	Description
EMAIL1	Enable and Disable	Allow for disabled/enabled status
EMAIL2	Server Configuration	Allow Email Server Configuration (port and host) at runtime
EMAIL3	Authentication Configuration	Allow SMTP authentication configuration if required at runtime
EMAIL4	Administrator Role	Allow for administration role for email (components X X and X)
EMAIL5	Repository Manager Role	Allow for repository manager role for email (components X X and X)
EMAIL6	Custom Role	Allow for custom role for email (custom components)

RSS Interface Functional Requirements

Covers the notification via RSS/Atom from an AONS installation.

Table 2.10. RSS Interface Functional Requirements

ID	Name	Description
RSS1	Query Log Messages	Must allow for querying of log messages

Search/Indexing Subsystem Functional Requirements

Covers the underlying search functionality which the primary modules utilise to save their information.

Table 2.11. Search/Indexing Subsystem Functional Requirements

ID	Name	Description
SEARCH1	Generic Searching	Must allow generic way for searching across all AONS domain objects
SEARCH2	Generic Indexing	Must allow generic way for indexing for all AONS domain objects
SEARCH3	Search Type Filtering	Must allow filtering to specific types of AONS domain objects (log messages, formats etc)

Http Subsystem Functional Requirements

The underlying subsystem which controls access to remove HTTP servers. This is mentioned since some installations of AONS may configure things like proxy servers and we need a single access point.

Table 2.12. Http Subsystem Functional Requirements

ID	Name	Description
HTTP1	Proxy Server Configuration	Must allow for configuration of a proxy at runtime
HTTP2	Proxy Authentication Configuration	Must allow for configuration of proxy authentication at runtime (username/password/realm)

Configuration Subsystem Functional Requirements

This is the essence of runtime configuration of AONS without needing to stop and modify XML.

Table 2.13. Configuration Subsystem Functional Requirements

ID	Name	Description
CONFIG1	Support Create	Allow creation of configuration objects
CONFIG2	Support Retrieval	Allow creation of configuration objects
CONFIG3	Support Update	Allow creation of configuration objects
CONFIG4	Support Delete	Allow creation of configuration objects
CONFIG5	Support Listing	Must provide listing of all current configurations

REST Interface Functional Requirements

This is the interface into AONS which is via REST.

Table 2.14. REST Interface Functional Requirements

ID	Name	Description
REST1	Repository CRUD	Support the full set of CRUD operations for Repository objects
REST2	Repository Listings	Support querying of available repositories
REST3	Repository Scan	Support RPC invocation of manually starting/cancelling a Repository scan

REST4	Registry CRUD	Support the full set of CRUD operations for Registry objects
REST5	Registry Listings	Support querying of available registries
REST6	Registry Synchronize	Support RPC invocation of manually starting/cancelling a Registry synchronize
REST7	Format Searching	Allow searching of all formats (Internal and External) via REST
REST8	Internal Format CRUD	Support the full set of CRUD operations for internal formats
REST9	Repository Scan Retrieval	Support the retrieval of repository Scans
REST10	Risk Summary	Support the retrieval of risk summary objects

Chapter 3. Non-Functional Requirements

Purpose

This section covers the non functional requirements. Any requirement which are not directly related to system behavior are specified here.

By their very nature, non-functional requirements are harder to define and test against, but we will specify them anyway as they give another perspective on the design of the system

Non-Functional Requirement Breakdown

Non functional requirements within AONS can be broken down like so:

Table 3.1. Non-Functional Requirements

High-Level Non-Functional Requirements	High-level non-functional requirements. These are also more "ideals" then set requirements.
System Non-Functional requirements	This area details non-functional requirements not directly related to business domain details but tasks which are necessary all the same.
Documentation Non-Functional requirements	This area deals with documentation non-functional requirements.
Registry Non-Functional Requirements	Non-functional requirements for the registry component of AONS.
Repository Non-Functional Requirements	Non-functional requirements for the repository component of AONS.
Logging Non-Functional Requirements	Non-Functional requirements for the logging component within AONS.
Task Non-Functional Requirements	Non-functional requirements for the task component within AONS.
Email Non-Functional Requirements	Non-Functional requirements for email within AONS.
RSS Interface Non-Functional Requirements	Non-Functional requirements for the RSS area within AONS.
Search/Indexing Subsystem Non-Functional Requirements	Non-functional requirements for the searching and indexing subsystem within AONS.
DAO Non-Functional Requirements	Details non functional requirements of the DAO Layer (REMEMBER TO LINK TO DAO IN GLOSSARY)
Configuration Subsystem Non-Functional Requirements	Non-functional requirements for the configuration subsystem within AONS
REST Interface Non-Functional Requirements	Non-functional requirements for the REST interface within AONS.

Business Managers Non-Functional Requirements	Non-Functional requirements for the business managers within AONS.

High-Level Non-Functional Requirements

High-level non-functional requirements. These are also more "ideals" then set requirements.

Table 3.2. High-Level Non-Functional Requirements

ID	Name	Description
NF-HL1	Platform Independent	Must be platform independent, probably written in Java
NF-HL2	Database Agnostic	Should be database agnostic, as much as possible, probably using an object/relational mapping tool or simple ANSI sql
NF-HL3	Flexible and Extensible	Must embody principals of flexibility and extensibility; with enough time could have "pluggin" style extensibility... but for now is should just be possible to extend it with new code + rewiring.
NF-HL4	Loosely Coupled	Must be loosely coupled
NF-HL5	Stable	Must be stable
NF-HL6	IDE independent	Must be IDE independent (should be possible to build/develop with any IDE)

System Non-Functional requirements

This area details non-functional requirements not directly related to business domain details but tasks which are necessary all the same.

Table 3.3. System Non-Functional Requirements

ID	Name	Description
SYS1	Provide Ability for Manual Static Reconfiguration	Provide deployers the ability to reconfigure the system with new non-functional requirements (new Data Repository Types, new external Registry Types and Obsolescence Rules).
SYS2	Monolithic GUI	Provide the ability to perform features through monolithic GUI.
SYS3	REST Interface	Provide the ability to utilise the system through REST interface.
SYS4	Persistence	Provide a simple and elegant persistence solution which allows

		for minimal programming and polymorphic object relations.
SYS5	Graphs	Provide a simple mechanism for displaying data in graphs
SYS6	UI Workflow	Provide an elegant mechanism of storing session state and managing UI workflows

Documentation Non-Functional requirements

This area deals with documentation non-functional requirements.

Table 3.4. Documentation Non-Functional Requirements

ID	Name	Description
DOC1	Open	The documentation must be written in an open format.
DOC2	Reusable	The documentation must be composable and make use of reuse to ensure we can create target guides fit for particular destinations.
DOC3	HTML	The documentation must be generated into standard HTML
DOC4	PDF	The documentation must be generated into PDF
DOC5	Build Information	The documentations should incorporate build and version information.

Registry Non-Functional Requirements

Non-functional requirements for the registry component of AONS.

Table 3.5. Registry Non-Functional Requirements

ID	Name	Description
NF-REG1	Table Locking	Must not lock database tables until the final update during a synchronize
NF-REG2	Adapters	Must allow creation of different types of Registries via an Adapter methodology

Repository Non-Functional Requirements

Non-functional requirements for the repository component of AONS.

Table 3.6. Repository Non-Functional Requirements

ID	Name	Description
NF-REP1	Table Locking	Must not lock database tables until the final update during a synchronize
NF-REP2	Adapters	Must allow creation of different types of Repositories via an Adapter methodology

Logging Non-Functional Requirements

Non-Functional requirements for the logging component within AONS.

Table 3.7. Logging Non-Functional Requirements

ID	Name	Description
NF-LOG1	Event Listeners	Messages should be created via listeners to business managers instead of hard coding calls to the logging subsystem (decoupling)
NF-LOG2	Resource Links	Logging subsystem should have a mechanism for finding resource links within the body of the subject/message text and ensuring that links go to the indended resource in the presentation layer. Each presentation layer will have to do this itself.

Task Non-Functional Requirements

Non-functional requirements for the task component within AONS.

Table 3.8. Task Non-Functional Requirements

ID	Name	Description
NF-TASK1	Event Listeners	Messages should be created via listeners to business managers instead of hard coding calls to the task subsystem (decoupling)
NF-TASK2	Resource Links	Task subsystem should have a mechanism for finding resource links within the body of the subject/message text and ensuring that links go to the indended resource in the presentation layer. Each presentation layer will have to do this itself.

NF-TASK3	Task Subclassing	Tasks should also be possible to have subclasses with extra information
NF-TASK4	Task Documentation	The set of tasks should be documented

Email Non-Functional Requirements

Non-Functional requirements for email within AONS.

Table 3.9. Email Non-Functional Requirements

ID	Name	Description
NF-EMAIL1	Event Listeners	Messages should be created via listeners to business managers instead of hard coding calls to the email subsystem (decoupling)
NF-EMAIL2	Resource Links	Email subsystem should have a mechanism for finding resource links within the body of the subject/message text and ensuring that links go to the indended resource in the presentation layer. Each presentation layer will have to do this itself.

RSS Interface Non-Functional Requirements

Non-Functional requirements for the RSS area within AONS.

Table 3.10. RSS Interface Non-Functional Requirements

ID	Name	Description
NF-RSS1	Support Formats	Must allow Atom 1.0 and RSS 2.0
NF-RSS2	Use If-Last-Modified	Must utilise If-Last-Modified query filter

Search/Indexing Subsystem Non-Functional Requirements

Non-functional requirements for the searching and indexing subsystem within AONS.

Table 3.11. Search/Indexing Subsystem Non-Functional Requirements

ID	Name	Description
NF-SEARCH1	Transactions	Must be transaction tied to the same overall business transactions are occuring

DAO Non-Functional Requirements

Details non functional requirements of the DAO Layer (REMEMBER TO LINK TO DAO IN GLOSSARY)

Table 3.12. DAO Non-Functional Requirements

ID	Name	Description
NF-DAO1	Abstract Database Implementation	Must hide details of database implementation from business layer (decoupled)
NF-DAO2	Transactional	Must be transactional; auto-commit is not an option
NF-DAO3	Use Templating	Should use templating methodologies (REMEMBER LINK TO TEMPLATING METHODOLOGY) to reduce code

Configuration Subsystem Non-Functional Requirements

Non-functional requirements for the configuration subsystem within AONS

Table 3.13. Configuration Subsystem Non-Functional Requirements

ID	Name	Description
NF-CONFIG1	Polymorphic	Allow arbitrary extension off base configuration object
NF-CONFIG2	Implementation Agnostic	Be agnostic towards subclasses of base configuration object. This will allow new configuration objects to be created without extending the configuration manager.

REST Interface Non-Functional Requirements

Non-functional requirements for the REST interface within AONS.

Table 3.14. REST Interface Non-Functional Requirements

ID	Name	Description
NF-REST1	Support REST	Utilise REST best practices (CRUD and error syntax) to expose business methods
NF-REST2	Support Some RPC	When the set of four CRUD operations (Create Retrieve

		Update and Delete) do not suffice, use RPC
NF-REST3	REST preferred	Should stick to CRUD ideals when possible and only use RPC when necessary (start a synchronize operation)
NF-REST4	Expose Business Managers	Must expose business manager non-functional requirements in an elegant manor
NF-REST5	Document Tasks	Must document the set of CRUD calls required to perform different business calls
NF-REST6	Document Each Resources	Must document all CRUD resources
NF-REST7	Per Resource Documentation	For every CRUD resource: • Must document allowable HTTP methods • Must document each parameter and it's function
NF-REST8	Document Each RPC Calls	Must document all RPC Calls

Business Managers Non-Functional Requirements

Non-Functional requirements for the business managers within AONS.

Table 3.15. Business Managers Non-Functional Requirements

ID	Name	Description
NF-BUS1	Database Agnostic	Separation of Business logic from database logic (decoupled)
NF-BUS2	Externally Demarcated Transactions	Must use externally demarcated transactions from business logic (decoupled)
NF-BUS3	Presentation Agnostic	Must be agnostic to invocation from (presentation) layers above (decoupled, allows for invocation by standard GUI or REST
NF-BUS4	Event Model	Should use an Event model (REMEMBER LINK TO EVENT MODEL) instead of hard coded dependencies between parallel systems
NF-BUS5	Temporal Pattern	Keep temporal/snapshots of various domain objects for statistical comparison

Chapter 4. Technologies

Introduction

In order to create a modern Java project, we utilise a wide plethora of technologies in order to get the job done. It is often said that with .Net there is only one way forward, which potentially means that the single one way is feature rich and well tested. The reality is often that .Net forces people to adopt a certain philosophy with few other options. Whilst there are certain benefits, it also means that competition between alteratives is minimal leading to stagnation. Java on the other hand is relatively "light" in it's most basic form and requires much augmentation by third party applications to build useful tools. Whilst this can lead to strengths in terms of what is available for usage, it also leads to an integration fist fight to get them all playing along nicely. In addition to the libraries needed to build the application, the development environment also often has a fair degree of customization to help when working with these tools.

This section will cover each of the main catagories of technology:

- Build and Development Tools
- Java Runtime Environment (JRE) and Java Development Kit (JDK)
- Libraries (both for building and deployment)
- Source Code Repositories
- Databases

Java Runtime Environment (JRE) and Java Development Kit (JDK)

AONS has been built using Java. This gives us many advantages, the main one being the "build one, run anywhere" mentality. Whilst it does have some restrictions on what one can expect from a platform, the end result is usually a very portable application.

In order to build AONS, one needs compatible version of the Java Development Kit (JDK). Within the the JDK, we implicitly require a version of the Java Runtime Environment (JRE). A discussion of each of these is beyond the scope of this document: if you're reading this you probably should be a fairly capable Java developer anyway. If you're just starting out, a good read of the Java philosophy of abstracting away from the underlying system is probably a good thing to do.

Within the AONS codebase, we make usage of many of the new features of the 1.5 JDK, namely generics and annotations. Development against a version of the JDK of 1.5/5.0+ is mandatory.

Java Development Kit (JDK)

Table 4.1. Technology Summary

Full Name	Java Software Development Kit
Version	1.5+ (otherwise known as 5.0)
Type	Java Development Kit

Home Page	http://java.sun.com

Description

The core development kit which we use to compile Java source files into platform independent bytecode files.

Java Runtime Environment (JRE)

Table 4.2. Technology Summary

Full Name	Java Runtime Environment (JRE)
Version	1.5+ (otherwise known as 5.0)
Type	Java Runtime Environment
Home Page	http://java.sun.com

Description

The runtime environment is the mechanism which allows platform independent bytecode to be run on an implementation platform.

Build and Development Tools

In order to perform development on AONS, you will need to be familiar with the build tool Ant and the Eclipse IDE.

Ant

Table 4.3.

Full Name	Ant
Version	1.6+
Type	Build Tool
Home Page	http://ant.apache.org

Description

Ant is a Java-based build tool. In theory, it is kind of like Make, without Make's wrinkles and with the full portability of pure Java code.

Eclipse IDE

Table 4.4. Technology Description

Full Name	Eclipse
Version	3.3 (Europa)
Type	Integrated Development Environment

Home Page	http://www.eclipse.org
License	Eclipse Public License (EPL) [http://www.eclipse.org/org/documents/epl-v10.php]

Description

Eclipse is a plugin based Integrated Development Environment (IDE). Eclipse comes with a set of standard plugins when downloaded which are further augmented based on the end users preferences.

Plugins Used

In addition to the base Java IDE install, we also have utilised the following plugins for our development:

- Subclipse
- Web Standard Tools
- J2EE Standard Tools Project
- Spring IDE

Subclipse

Table 4.5. Technology Description

Full Name	Subclipse
Version	1.2.3
Type	Eclipse Plugin
Home Page	http://subclipse.tigris.org/

Description

Subclipse is an Eclipse Team Provider plug-in providing support for ??? within the Eclipse IDE IDE.

Web Standard Tools

Table 4.6. Technology Summary

Full Name	Web Standard Tools
Version	2.0.0.v200706041905
Type	Eclipse Plugin
Home Page	http://www.eclipse.org/webtools/wst/main.php

Description

The project vision is to extend eclipse platform with support for building multi-tier Web applications. The project will grow the community of eclipse users, and grow the community of developers that create Web applications based on Open Standards and Technologies. In this way, we will help make eclipse the industry standard tool integration platform for development based on Open Standards and Technologies. The project must preserve the eclipse value proposition by providing integration, ease-of-use, function, and "coolness". Web artifacts must be first class citizens with respect to the capabilities that eclipse

users expect. Servers must be first class execution environments, including both Open and Commercial implementations, therefore encourage support of eclipse by server vendors.

J2EE Standard Tools Project

Table 4.7. Technology Summary

Full Name	J2EE Standard Tools Project
Version	2.0.0.v200706110805
Type	Eclipse Plugin
Home Page	http://www.eclipse.org/webtools/jst/main.php

Description

The project vision is to extend eclipse platform with support for building multi-tier J2EE applications. The project will grow the community of eclipse users, and grow the community of developers that create Web applications based on the J2EE platform. In this way, we will help make eclipse the industry standard tool integration platform for development based on Open Standards and Technologies. The project must preserve the eclipse value proposition by providing integration, ease-of-use, function, and "coolness". Web artifacts must be first class citizens with respect to the capabilities that eclipse users expect. Servers must be first class execution environments, including both Open and Commercial implementations, therefore encourage support of eclipse by server vendors.

Spring IDE

Table 4.8. Spring IDE

Full Name	Spring IDE
Version	2.0.0.v200706271108
Type	Eclipse Plugin
Home Page	http://springide.org/blog/

Description

Spring IDE provides support features within the Eclipse IDE platform for ??? development. It gives you useful tools to validate and visualize your bean definitions as well as support while editing Spring Bean defintions with content assist and much more.

Building AONS with Eclipse

AONS is developed with the Eclipse IDE, but not entirely built with it. The Eclipse IDE may automatically compile classes as they are developed, but to build the final Java Web Archive (WAR file) and the deployment release, we use Ant

Chapter 5. Building AONS

AONS Ant build tasks

With every Ant build file, there are available a set of tasks which enable use to build the application. Even though AONS is developed with Eclipse, it essentially can be built standalone via Ant. AONS has two build files, one for building the application from source and another for deploying an almost complete build package. The second stage is useful for people developing AONS and the second is useful for those deploying a pre-made distribution copy.

Development Build Tasks

The following tasks and their descriptions are available in the development build tasks:

Table 5.1. Development Build Tasks

Task Name	Task Description
build-archive	builds the java archive file from the binary class files
clean	removes all generated artifacts
compile	builds the source code files (source and test)
create-war	creates a war file for deployment
test	runs junit test files over AONS java classes
deploy	deploys the built war file to the deployment directory
java-doc	create the Javadoc files based on the source code
create-release-package	builds the release package for installation

Distribution Build Tasks

The distribution build tasks are currently just "build-war" and "deploy". This is because the distribution package is already almost 100% built and is essentially just taking on board the final site specific configuration parameters. The following guide should be followed when building:

Build

In order to build the AONS web archive

Creating Database Schema

AONS does not need an explicit DDL creation step on initial installation due to ??? creating DDL on application startup. Most of the time this will be viewed as a positive, but if for some reason this is not desired, it can be explicitly turned off in the Spring configuration.